COMP534 Assignment 2

Kok Wing, Law (201629894)

Q1. A Section explaining the project (10%), including (but not limited to):

- Introduction of the dataset

The balanced EMNIST (Extended MNIST) dataset includes a mixture of numbers and upper and lower case letters. The dataset contains a total of 47 classes, each class having the same number of data. There is a total of 131,600 data, in which 112,800 is the training and 18,800 is the testing data.

- Describe the structure of your MLP and CNNs, and discuss how you build them up.

```
NeuralNetwork(
  (activation_function): ReLU()
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_actfun_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=512, bias=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=47, bias=True)
  )
)
```

My MLP contains an input layer, 3 hidden layers and an output layer. In the input layer, 784 features (flatten 28*28 pixels) are input then passed to the next 3 hidden layers, each composed of 512 nodes with a bias and an activation function. The output layer contains 47 outputs, which refers to the 47 classes in the dataset.

```
CNN(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=2000, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=47, bias=True)
)
```

My CNN model contains 2 convolutional layers, 1 max pooling, 1 fully-connected hidden layer and output layer. The first convolutional layer has 10 kernels with a kernel size of 5*5, no padding and 1 stride. The second convolutional layer has 20 kernels with kernel size 3*3, no padding and 1 stride. The pooling layer is placed after the first

convolutional layer, in the size of 2*2 and 1 stride. After flatten the output from the second convolutional layer, it passes to the hidden layer with 500 nodes, bias and activation function. Then the output layer outputs 47 nodes and returns with a softmax function.

Q2. A Section explaining the rationale of your design on MLP and CNNs (25%), including (but not limited to):

- What kind of hyperparameters or techniques do you choose to tune and explore

  - ❖ Adaptive Learning Rate (0.01, 0.1)
  - ❖ Activation function (ReLU, Leaky ReLU, ELU)
  - ❖ Optimizers (SGD, ADAM, AdaGrad, ASGD)
  - ❖ Batch Normalization (explore two options: with Batch normalization, without Batch normalization)
  - ❖ L1 & L2 regularization (explore three options: without L1&L2 regularization, with L1 regularization, with L2 regularization)
  - ❖ Dropout (explore two options: with or without Dropout)

MLP (epochs = 5)

Learning Rate	Activation Function	Optimizer	Batch Normalization	Regularization	Dropout	**Accuracy**
0.01	ReLU	SGD	No	No	No	62.38
0.1						82.81
	LeakyReLU					83.46
	ELU					81.82
		Adagrad				70.31
		ASGD				84.26
			Yes			84.95
				L1		71.91
				L2		85.05
					Yes	84.64

CNN (epochs = 5)

Learning Rate	Activation Function	Optimizer	Batch Normalization	Regularization	Dropout	**Accuracy**
0.01	ReLU	SGD	No	No	No	79.40
0.1						86.20
	LeakyReLU					86.35
	ELU					86.22
		Adagrad				82.96
		ASGD				86.38
			Yes			87.49
				L1		81.77
				L2		87.70
					Yes	86.37

My best MLP & CNN model (both have the same result):

Adaptive Learning Rate: *0.1*

Activation function: *Leaky ReLU*

Optimizers: *ASGD*

Batch Normalization: *with Batch normalization*

L1 & L2 regularization: *with L2 regularization*

Dropout: *without Dropout*

- Why do you choose those hyperparameters or techniques

Learning rate determines the steps and epochs to be optimized, activation function return the output within a range, optimizer improve the performance of achieving the weight and bias in the training process, batch normalization reduce the effect of extreme value of weight, regularization prevents overfitting to the training data and dropout could improve the accuracy.

- For each technique, please describe how it affects your model's performance and analyze the reasons why the technique can boost or decrease the performance of your models.

Learning Rate

As I have chosen relatively small epochs (i.e. 5, due to the long runtime), a higher learning rate would adjust the weight and bias in a larger step and can be optimized within limited epochs.

Activation Function

The Leaky ReLU has a small constant gradient slope on the negative side, which can output negative values compared to the ReLU function, and has a constant negative rate compared to ELU.

Optimizer

The Adagrad gives the worst performance because it would reduce the learning rate as the training step increases. In the case with small epochs, the Adagrad would make the training process less significant. The SGD has a better result than Adagrad, but still not the best among the three optimizers. The SGD calculates the loss function with random data, which causes a fluctuated loss and accuracy throughout the epochs. With a small number of epochs, it could easily obtain a less optimized result than the others. The ASGD improves the SGD by averaging the weight in each iteration. It makes the training process less sensitive to loss from the random data.

Batch Normalization

Batch normalization applies a mean and standard deviation to each layer's output to prevent the weight of certain nodes being much larger than the others. By normalizing each batch, the accuracy would increase.

Regularization

The L2 regularization adds a penalty to the loss function to reduce the complexity of the model and prevent overfitting. Compared to L1 regularization, L2 takes the square of the coefficient instead of the absolute value, which is less sensitive to the outliers.

Dropout

Some random output from the previous layer would be ignored, which could be quite destructive to the model.

- Do you ever have overfitting or underfitting issue while training the models? How do you deal with overfitting or underfitting issue?

I have an underfitting issue on the MLP model with parameters of 0.01 learning rate. It has only around 60% on both training and testing data. The issue is improved by increasing the learning rate from 0.01 to 0.1. Moreover, I have an overfitting problem when training the CNN model, in which the train accuracy increases while the test accuracy decreases. One of the possible solutions is to increase the lambda value in the L2 regularization.

Q3. A section describing and explaining your results (10%), including (but not limited to):

- Describe and explain the training loss, testing loss and testing accuracy of MLP and CNNs.

MLP:

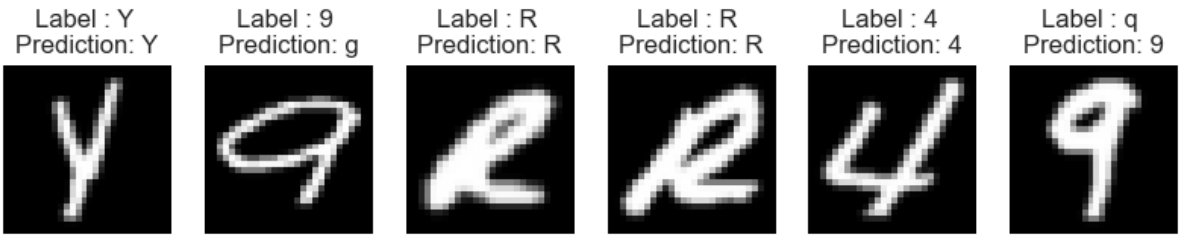
```
Average Training Loss: 1.4823
Average Test Loss: 0.5012
Average Training Acc: 83.084
Average Test Acc: 83.241
```

CNN:

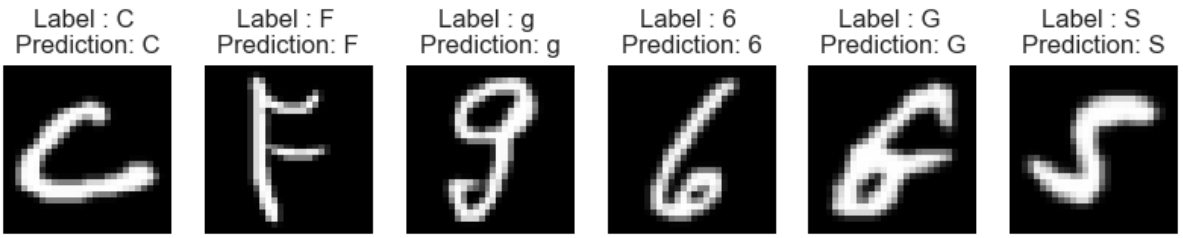
```
Average Training Loss: 0.7761
Average Test Loss: 0.3852
Average Training Acc: 87.535
Average Test Acc: 86.935
```

- Describe and explain the predicted results of your models.

MLP:



CNN:



The above lists 6 examples of the prediction from MLP and CNN. The prediction results of MLP cannot distinguish between “9” and “g”, and also “q” and “9”. CNN can successfully classify all the 6 examples with correct labels, even for the letter “g” which cannot distinguish with “9” in the MLP.

- Your analysis of the performance of MLP and CNNs based on your own AI knowledge. You need to discuss the pros and cons of MLP and CNNs and analyse why a certain model is better than another model.

As MLP is a feed-forward and fully connected model, it is capable of solving nonlinear problems such as binary classification problems. However, when MLP encounters the data with large amounts of parameters such as large size images, the process would be relatively slow because all nodes in each layer are fully connected to the next layer and all the weight and bias need to be computed in the training process. Furthermore, MLP cannot retain the spatial information when the image is flattened as the input of the MLP.

Both problems can be solved by the CNN model. The nodes in CNN are not fully connected and the weights are shared. It can process images faster than MLP. More importantly, CNN contains convolutional layers which are formed by kernels to downsample the image and retrieve the spatial information. Objects can be detected in every position of the image. Also, the pooling layer can extract the feature and retain its position with less image size. With a lower dimension, the CNN can compute faster than the MLP.

Q4. The final conclusions (5%), including (but not limited to):

- Your own reflection on this project, for example, what you have learned via this project, which part you could do better next time, etc.

I have learnt how to figure out the best hyper-parameter of a model and understand the differences and advantages between the MLP and CNN model. As I do the hyper-parameter exploration purely in manuals at this time, it takes me a very long time to go through different models. The exploration process could be improved by making it automatic to save the runtime of different models.